



[40101/08201] *AP* *AS* *EW*

PATENT THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s) : Nunoe
Serial No. : 09/920,995
Filing Date : August 1, 2001
For : System and Method for Implementing a Smart System Call
Group Art Unit : 2194
Examiner : Charles E. Anya

Mail Stop: Appeal Brief-Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Mailing
I hereby certify that this correspondence is being deposited with
U.S. Postal Services as first class mail in an envelope addressed
to:

Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

By: *MM* Date: January 17, 2006
Michael J. Marcin, (Reg. No. 48,198)

TRANSMITTAL

In response to the Notice of Appeal filed August 17, 2005 and the Advisory Action dated June 20, 2005, transmitted herewith please find an Appeal Brief (in triplicate) for filing in the above-identified application. Applicants request a two-month extension. Please charge the Credit Card of **Fay Kaplun & Marcin, LLP** in the amount of \$950.00 (PTO-Form 2038 is enclosed herewith). The Commissioner is hereby authorized to charge the **Deposit Account of Fay Kaplun & Marcin, LLP NO. 50-1492** for any additional required fees. A copy of this paper is enclosed for that purpose.

Respectfully submitted,

By: *MM*
Michael J. Marcin, Reg. 48,198

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel: (212) 619-6000
Fax: (212) 619-0276

Dated: January 17, 2006

01/23/2006 WABDEL1 00000052 09920995
02 FC:1252 450.00 OP



PATENT
Attorney Docket No.: 40101 / 08201

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of:)
)
 Hdei Nunoe)
)
 Serial No.: 09/920,995) Group Art Unit: 2194
)
 Filed: August 1, 2001) Examiner: Charles E. Anya
)
 For: SYSTEM AND METHOD FOR)
 IMPLEMENTING A SMART)
 SYSTEM CALL) **Board of Patent Appeals and
 Interferences**

Mail Stop: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

In support of the Notice of Appeal filed August 17, 2005, and pursuant to 37 C.F.R. § 41.37, Appellant presents their appeal brief in the above-captioned application.

This is an appeal to the Board of Patent Appeals and Interferences from the Examiner's final rejection of claims 1-7 in the final Office Action dated March 18, 2005. The appealed claims are set forth in the attached Claims Appendix.

01/23/2006 WABDEIR1 00000052 09920955
500.00 IP
01 FC:1402

1. Real Party in Interest

This application is assigned to Wind River Systems, the real party in interest.

2. Related Appeals and Interferences

There are no other appeals or interferences which would directly affect, be directly affected, or have a bearing on the instant appeal.

3. Status of the Claims

Claims 1-7 have been rejected in the final Office Action. The final rejection of claims 1-7 is being appealed.

4. Status of Amendments

All amendments submitted by the appellant have been entered. None were submitted after the Advisory Action.

5. Summary of Claimed Subject Matter

The present invention relates to a system and method for implementing a “smart system call” that is capable of making function calls to privileged memory locations. Specifically, the present invention describes a method for providing either direct or indirect access to a software function by determining the current processing mode of an executing software function. (See Specification, p. 5, ll. 12-20). The processing mode is defined as either

an unprivileged processing mode of protection having a lower level access right to the memory of a computer system, or a privileged processing mode of protection having a higher level access right to the memory. (See Id.).

The “smart system call” includes a code stub that may be used to determine the current mode of execution and determine where the current mode is privileged (i.e., allowed direct access to the desired memory address). (See Id., p. 5, l. 23 – p. 6, l. 10). If the current mode is unprivileged, an alternate indirect system call mechanism may be used to allow for indirect access to the desired memory address. (See Id.). This mechanism may be an interrupt or exception based system call that may cause the spawning of privileged mode tasks in order to execute the desired function. (See Id.).

According to an exemplary code stub of a preferred embodiment of the present invention, if a current code is not unprivileged, then a direct jump (or change in program control) is made to the entry address of the desired function. (See Id., p. 8, ll. 6-14). Conversely, if the current mode is unprivileged, then the system call is made using the interrupt based system call. (See Id.). Therefore, the present invention may allow for easy implementation of direct access to protected memory in an operating system that uses interrupt-based system calls. (See Id., p. 8, ll. 16-24). In other words, this exemplary code stub may be used to emulate exception-based system calls in an operating system that does not support such calls.

6. Grounds of Rejection to be Reviewed on Appeal

I. Whether claims 1-7 are unpatentable under 35 U.S.C. § 103(a) as obvious over U.S. Patent No. 6,529,985 to Deianov et al. ("the Deianov patent") in view of U.S. Patent No. 6,658,571 to O'Brien et al. ("the O'Brien patent").

7. Grouping of Claims

Claims 1-7 may stand or fall together.

8. Argument

I. The Rejection of Claims 1-7 Under 35 U.S.C. § 103(a) as Being Obvious Over U.S. Patent No. 6,529,985 to Deianov et al. in View of U.S. Patent No. 6,658,571 to O'Brien et al. Should Be Reversed.

A. The Examiner's Rejection

In the final Office Action, the Examiner rejected claims 1-7 Under 35 U.S.C. § 103(a) as being unpatentable over the Deianov patent in view of the O'Brien patent. (See 03/18/05 *Office Action*, p. 2, ll. 9-10).

The Deianov patent describes a system for the selective interception of system calls, wherein a system call wrapper is used to execute in process address space of selected processes in a computer memory. (See the Deianov patent, col. 5, ll. 55-63). The computer memory includes operating system address space as the location for the execution of an operating system kernel loaded with an interception model. (See *Id.*, col. 5, l. 66 – col. 6, l. 2).

The system further includes operating system pointers to system calls, which is located in an operating system interrupt vector table. (See Id., col. 6, ll. 5-6). The system creates a copy of the pointers to each system call that is to be intercepted and saves the copies in the operating system address space (or, alternatively, in the user address space). (See Id., col. 6, ll. 16-20). These pointers to the to-be-intercepted system calls are replaced with pointers to the interception module, so that the interception module will execute the to-be-intercepted system call. (See Id., col. 6, ll. 20-26). Thus, when a call is made to a to-be-intercepted system call, the operating system uses the pointer in the interrupt vector table to the interception module to execute the interception module. (See Id., col. 6, ll. 35-38). The interception module maintains an association table of identifiers of selected processes and system call wrapper entry points. (See Id., col. 7, ll. 9-15). When the interception module executes in response to the calling of a system call, the interception module can determine if the calling process is associated with a system call wrapper. (See Id., col. 7, ll. 21-24).

The O'Brien patent describes a security framework for dynamically wrapping software applications to limit corruption or damage caused to an operating system. (See the O'Brien patent, col. 2, ll. 12-16). This security framework is capable of providing application-based controlled access to computing resources such as memory, files, network sockets, and processes. (See Id., col. 3, ll. 2-9). Specifically, the framework includes kernel-loadable security modules that make and enforce application-specific and resource-specific policy decisions for the applications. (See Id., col. 3, ll. 38-55). Therefore, the security modules may grant or deny access to computing resource on the basis of either the application making the request or the

resource being requested. (See Id.). Thus, each security module is able to wrap one or more application and prevent unauthorized applications from accessing a computing resource in the event the application executes malicious software. (See Id.). In other words, the security framework described in the O'Brien patent is implemented in order to determine whether or not a specific application may be granted access or denied access to a given computing resource.

B. The Cited Patents Do Not Disclose When the Current Processing Mode is an Unprivileged Processing Mode, Executing an Indirect Program Flow Control Instruction to Cause Execution of the Instruction within Software Having the Privileged Processing Mode and Determining a Current Processing Mode of an Executing Software, and as Recited in Claim 1.

The Examiner asserts that the Deianov patent teaches determining whether the executing software function can access the system resource directly or indirectly. (See 03/18/05 *Office Action*, p. 2, ll. 12-21). Since the Deianov patent fails to address the protection level of the access to the system resource, the Examiner's claim is unfounded. Thus, it is respectfully submitted that neither the Deianov patent nor the O'Brien patent teach or disclose "when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within software having the privileged processing mode" as also cited in claim 1.

As discussed above in the Summary of Claimed Subject Matter, an exemplary method of the present application is used to determine whether direct access or indirect access to a software function is provided based on the current processing mode of an executing software function. A processing mode that is privileged has a higher level access right to memory and

thus, is sufficiently trustworthy to be provided with direct access to the desired memory access.

Alternatively, a processing mode that is unprivileged has a lower level access right to memory and is not sufficiently trustworthy. Thus, this mode is only provided with indirect access to the desired memory access. Regardless of whether the processing mode is privileged or unprivileged, the software function will still have access to the requested computing resource. The only determination being made is whether to provide direct or indirect access. Thus, the purpose of the examination of the current processing mode as recited in claim 1 is for selecting the type of the access a requesting software function may obtain.

In contrast to determining whether a processing mode has either a higher or lower level access right to the memory, the Deianov patent describes the use of an interception module for the selective execution of either system calls or system call wrappers. As discussed above, the interception module replaces a pointer in an interrupt vector table to a system call with a pointer to a system call wrapper to be executed instead of the system call. The determination made by the interception module of the Deianov patent is merely for the purpose of either executing the process or selecting an associated system call to execute the process. A select process, which is to intercept system calls, is not equivalent to either to having a higher or lower access right to memory. Additionally, a process that makes a system call, which is not one of the selected processes to intercept system calls, is not equivalent to having a higher or lower access right to memory. In other words, whether or not a process has an associated system call wrapper is neither similar nor equivalent to operating in a privileged or unprivileged processing mode.

According to the Deianov patent, the interception module determines whether a system call was made by a system wrapper through the use of an association table of process identifiers of select processes and system call wrapper entry points. (See the Deianov patent, col. 7, ll. 9-15). As discussed above, when a process makes a system call the pointer to which has been replaced with a pointer to the interception module, then the interception module will execute the process. Selected processes (within the association table) are loaded by a modified loader program, wherein such processes intercept system calls. (See Id., col. 7, ll. 42-49). Alternatively, non-selected processes (not within the association table) are loaded with the default operating system loader. (See Id.) Since the non-selected processes do not have associated system call wrappers, the non-selected processes do not intercept system calls. (See Id.). When a process makes a system call to be intercepted, the interception module executes. (See Id., col. 8, ll. 12-28). For the selected processes (having system call wrappers), the interception module will execute the system call wrapper for the process. (See Id.). While for non-selected processes, the interception module will execute the system call for the process. (See Id.). The disclosure for the Deianov patent fails to make any reference to providing unprivileged processing modes (or applications with lower level access rights to memory) with access to privileged memory locations.

Furthermore, the Examiner appears to be equating the interception module's execution of the system call wrappers for selected processes having system call wrappers with the execution of "a indirect program flow control instruction to directly access an instruction within software having the privileged processing mode" of claim 1. (See 03/18/05 *Office Action*,

p. 2, ll. 12-21; and 06/20/05 *Advisory Action*, p. 2, ll. 1-8). It is important to note that the present invention allows simple implementation of direct access to *protected* memory in systems that employ interrupt-based system calls, thereby providing the flexibility to change protection modes dynamically. As opposed to the use of an interception module executing system call wrapper for selected processes and executing system calls for non-selected processes of the Deianov patent, the present invention may be implemented to run applications while using a protectionless memory. As discussed above, the “smart system call” allows for unprivileged applications to make function calls to privileged memory locations. The method according to Deianov simply provides for the selective interception of system calls to insure that the system call wrapper only executes when a select process makes a system call. The default system call is executed when the system call is made by a non-selected process. The interception module provides nothing more than a means for making a system call for a process or executing a system call wrapper on the basis of the association table of selected processes, wherein the system call wrapper is loaded into the process address space of the calling process. This method does not permit access to privileged memory resources for unprivileged processes. Thus, the Deianov patent fails to teach or suggest “when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within software having the privileged processing mode” as recited in claim 1. The Examiner’s claim in the comments of the 06/20/2005 *Advisory Action* that states Deianov teaches determining whether the executing software function can access the system resource directly or indirectly are futile since Deianov fails to teach or suggest providing privileged access to an unprivileged process.

In addition, the Examiner acknowledged that the Deianov patent fails to teach or suggest “determining a current processing mode of an executing software function” as recited in claim 1. (See 03/18/05 *Office Action*, p. 3, ll. 1-5). However, the Examiner further stated that the O’Brien patent shows these claimed elements, thereby rendering the claimed subject matter obvious over the Deianov patent. (See *Id.*). Appellant respectfully disagrees with the Examiner’s rejection of claim 1.

It is respectfully submitted that O’Brien neither discloses nor suggests “determining a current processing mode of an executing software function” and “when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within software having the privileged processing mode,” as recited in claim 1. In O’Brien, the security module enforces:

[A] set of rules identifying which computing resources 106 the browser is allowed to access as well as what permissions the browser has. If there is no rule that allows access to a computing resource 106, then the security framework 101 refuses any requests to access that computing resource 106.

(See O’Brien, col. 7, ll. 27-33). Reading preset rules or permissions is not a determination of a current processing mode. In the present invention, the examination of the “current processing mode” is used to decide whether an executing software function should be allowed to access a system resource directly or indirectly and **not** whether it is allowed to access the resource. The software functions of the present invention are presumed to have access to system resources. Thus, the present invention allows the computing system to decide the type of access that the software functions will obtain.

In further support of the rejection, the Examiner states that Deianov teaches determining whether the executing software function can access the system resource directly or indirectly, and, O'Brien, therefore, is being cited solely to show the step of determining the access level of the executing software function to a system resource. (See 3/18/05 *Office Action*, p. 2, ll. 12-21 and p. 6, ll. 7-20). Applicant respectfully disagrees with the Examiner's reading of the Deianov patent and the O'Brien patent, and the combination of their disclosures.

Initially, it is respectfully submitted that the Deianov patent does not disclose or suggest determining whether the executing software function is to be intercepted. As discussed above, the Deianov patent only refers to system calls that are to be intercepted, and those that are not intercepted. In contrast to the Examiner's assertion, it is respectfully submitted that at no point does the Deianov patent disclose how system calls are distinguished between intercepted and not-intercepted. In fact, the Deianov patent describes the method of a process executing a system call as follows:

Executing processes make system calls 115. When a process makes a system call 115 *that is to be intercepted*, the interception module 111 executes. The interception module 111 examines the association table 127 to determine whether the process that made the system call 115 is associated with a system call wrapper 125.

(See Deianov, col. 8, ll. 14-19) (emphasis added). Thus, the interception module examines only system calls that are intercepted. The Deianov patent does not disclose or suggest how to distinguish between system calls which are to be intercepted and those that are not intercepted, or what happens to the system calls that are not intercepted.

It is respectfully submitted that neither the Deianov patent nor the O'Brien patent, either alone or in combination, disclose or suggest "determining a current processing mode of an executing software function" and "when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within software having the privileged processing mode," as recited in claim 1. Because claims 2-5 depend from, and, therefore include all of the limitations of claim 1, it is respectfully submitted that these claims are also allowable.

Claim 6 includes substantially the same limitations as claim 1, including "determining a current processing mode of the program code segment" and "execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 6 is also allowable.

Claim 7 includes substantially the same limitations as claim 1, including "determining a current processing mode of the program code segment" and "execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 7 is also allowable.

9. Conclusions

For the reasons set forth above, Appellant respectfully requests that the Board reverse the final rejections of the claims by the Examiner under 35 U.S.C. § 103(a), and indicate that claims 1-7 are allowable.

Respectfully submitted,

Date: January 17, 2006

By: 
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel: (212) 619-6000
Fax: (212) 619-0276

CLAIMS APPENDIX

1. A method, comprising:
 - determining a current processing mode of an executing software function;
 - when the current processing mode is a privileged processing mode, executing a direct program flow control instruction to directly access an instruction within software having the privileged processing mode; and
 - when the current processing mode is an unprivileged processing mode, executing an indirect program flow control instruction to cause execution of the instruction within software having the privileged processing mode.
2. The method of claim 1, wherein the direct program flow control instruction is a jump instruction.
3. The method of claim 1, wherein the indirect program flow control instruction is an interrupt instruction.
4. The method of claim 1, wherein the software having the privileged processing mode is operating system software.
5. The method of claim 4, wherein the software having the privileged processing mode is kernel software.

6. A method, comprising:

identifying a program code segment implementing an access to a memory area to be executed within a privileged processing mode;
replacing the program code segment with a substitute code segment;
wherein the substitute code segment includes program code to
identify a current processing mode of the program code segment,
execute a direct program flow control instruction if the current processing mode is the privileged processing mode, and
execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode.

7. A software application, comprising:

software code implementing application functionality; and
a smart system call into an operating system;
wherein the smart system call comprises software code to
identify a current processing mode of the program code segment,
execute a direct program flow control instruction if the current processing mode is the privileged processing mode, and
execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode.

EVIDENCE APPENDIX

No evidence has been entered or relied upon in the present appeal.

RELATED PROCEEDING APPENDIX

No decisions have been rendered regarding the present appeal or any proceedings related thereto.